JSON Analyse

JSON Bibliothek

Da erst mit Java 9 eine native Unterstützung von JSON Dateien verfügbar ist, muss eine Bibliothek zum erstellen und parsen von JSON Dateien, in das Projekt eingebunden werden. Da an diesem Projekt viele Entwickler mitwirken werden, ist das erste Auswahlkriterium die Popularität der Bibliotheken. Die Popularität ist der Liste der Top 100 Java Bibliotheken[1] entnommen, demnach ist Jackson[2] die beliebteste Bibliothek, dicht gefolgt von GSON[3]. Als vertreter der hinteren Plätze wurden JSON-P und JSON.simple ausgewählt.Auf dem dritten Platz findet sich json.json[4] und auf dem vierten JSON.simple[5].

Vorteile der einzelnen Bibliotheken

Jackson	GSON	JSON-P	JSON.simple
Hohe Performance[6] bei großen Dateien	Hohe Performance bei kleinen Dateien		Mittelmäßige Performance bei großen und kleinen Dateien
Bietet Möglichkeit 2 der Implementierung an siehe JSON Implementierung	Bietet Möglichkeit 2 der Implementierung an siehe JSON Implementierung		

1 https://blog.takipi.com/the-top-100-java-libraries-in-2016-after-analyzing-47251-dependencies/
2 https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind
3 https://mvnrepository.com/artifact/com.google.code.gson/gson
4 https://mvnrepository.com/artifact/org.json/json
5 https://mvnrepository.com/artifact/com.googlecode.json-simple/json-simple
6 https://blog.takipi.com/the-ultimate-json-library-json-simple-vs-gson-vs-jackson-vs-json/

Nachteile der einzelnen Bibliotheken

Jackson	GSON	JSON-P	JSON.simple
Niedrige Performance[7] bei kleinen Dateien	Niedrige Performance bei großen Dateien	Niedrige Performance bei großen und kleinen Dateien	

Für dieses Projekt wird eine Bibliothek benötigt die möglichst schnell kleine (<100kB) Dateien speichern kann. GSON sticht hierfür besonders hervor da es in dieser Disziplin die beste der verglichenen Bibliotheken ist. Dass GSON die 2. Möglichkeit der Implementierung (siehe nächstes Kapitel) anbietet spielt für diese Entscheidung keine Rolle, da dies nicht benötigt wird (siehe nächstes Kapitel). Beim Aufwand der Implementierung der verschiedenen Bibliotheken konnte kein Unterschied festgestellt werden, weshalb dies ebenfalls keine Auswirkung auf die Entscheidung hat. Als einziges Kriterium mit Aussagekraft ist die Performance das entscheidende und somit fällt die Wahl der JSON Bibliothek auf GSON.

JSON Implementierung

Es gibt grundsätzlich zwei Möglichkeiten Java Objekte in JSON umzuwandeln. Im folgenden werden die Vor- und Nachteile beider Möglichkeiten analysiert.

1. Möglichkeit speichern der wichtigen Attribute

Vorteile	Nachteile
Geringer Speicherbedarf Bessere Wartbarkeit	Mehr Implementierungsaufwand

2. Möglichkeit speichern aller Attribute

Vorteile	Nachteile
Geringer Implementierungsaufwand Speichern von unbekannten Objekten	Schlechte Wartbarkeit Hoher Speicherbedarf

7 https://blog.takipi.com/the-ultimate-json-library-json-simple-vs-gson-vs-jackson-vs-json/

Möglichkeit 2 bietet dem Entwickler an unbekannte Objekte zu speichern, dies ist für dieses Projekt allerdings nicht notwendig. Somit spricht nur der geringere Implementierungsaufwand für diese Möglichkeit. Die bessere Wartbarkeit der 1. Möglichkeit ist der ausschlaggebende Aspekt weshalb wir uns für diese entschieden haben.

Wahl der wichtigen Attribute

Um nur die wichtigen Attribute abzuspeichern muss festgelegt werden welche dies sind, hierbei wird zwischen Konfiguration und Zustand unterschieden.

- **Konfiguration**:

Objekt	Attribute
Stationen	Name, Kürzel, direkte Vorgänger, X/Y Koordinaten
Kreuzungen	Name, direkte Vorgänger, X/Y Koordinaten
Fließbänder	Position der Eckpunkte
Netzwerkverbindung zum Controller	IP-Adresse, Portnummer

- **Zustand:**

Objekt	Attribute
Stationen	Name, Kürzel, direkte Vorgänger, X/Y Koordinaten, anstehende Schlitten
Kreuzungen	Name, direkte Vorgänger, X/Y Koordinaten
Fließbänder	Position der Eckpunkte
Netzwerkverbindung zum Controller	IP-Adresse, Portnummer
empfangene Befehle	Befehl-Code, Abarbeitungs-Zustand